



Flux Alerts in the PhotPipe

prepared by: L. Wyrzykowski and S. T. Hodgkin
reference: GAIA-C5-TN-IOA-LW-002
issue: D
revision: 3
date: 2009-07-24
status: Draft

Abstract

This document describes Flux Alert system, part of the CU5 photometric data processing. Also, the Data Model for DU17 is described. The main purpose of this document is to describe the Java code in high details.

This document is supersede by the set of documents on each taker, e.g. GAIA-C5-TN-IOA-LW-003 and is only valid until Rev. 98126.

Document History

Issue	Revision	Date	Author	Comment
D	4	2009-07-24	LW	Updated to Rev.98126.
D	3	2009-07-20	LW	Detector and Curator modified. New class, Classifier described.
D	2	2009-06-08	LW	Major changes to Updater and Detector
D	1	2009-03-20	LW	Alert Table fields modified
D	0	2009-02-26	LW	Creation

Contents

1	Introduction	6
1.1	Objectives	6
1.2	Scope	6
1.3	Assumptions/Limitations	6
2	FluxAlertDetector	6
2.1	Taker	6
2.1.1	Data Streams	6
2.1.2	Setup	7
2.1.3	Take	7
2.1.4	Teardown	7
2.2	FluxAnomalyDetector	7
2.2.1	<i>checkTransit</i> method	7
2.2.2	<i>checkForAnomaly</i> method	8
2.2.3	CreateAlert	8
2.3	Detection Algorithm	9
2.3.1	Setup	9
2.3.2	Detect	10
2.3.3	checkForAnomaliesOnCcds	11
2.3.4	isDeviant	11
2.3.5	areDeviationsOfSameSign	11

3	FluxAlertUpdater	11
3.1	Taker	12
3.1.1	Data Streams	12
3.1.2	Setup	12
3.1.3	Take	12
3.1.4	LoadTransits	12
3.1.5	Teardown	13
4	FluxAlertCurator	13
4.1	Taker	13
4.1.1	Data Streams	13
4.1.2	Setup	13
4.1.3	Take	13
4.1.4	Teardown	13
4.2	Classifier - classification of alert	14
4.2.1	checkAnomalyInSecondFov	16
5	Alert Table	16
6	DU17's Data Model	16
6.1	AlertFovTransitDt	16
6.1.1	AlertFovTransit	17
6.2	AlertSourceDt	18

6.2.1	AlertSource	19
6.3	FluxAlertCandidateDt	21
6.3.1	FluxAlertCandidate	21

1 Introduction

1.1 Objectives

This document contains details on Flux Alerts system designed by DU17, which will operate as a part of CU5 photometric pipeline. The aim of the system is to analyse the ingested data in order to detect new objects and variable stars changing their brightness dramatically. The system issues alerts to the general astronomical community to encourage detailed follow-up of these objects.

1.2 Scope

The work flow of Flux Alerts system is divided into three Data Takers (CITATION):

FluxAlertDetector is responsible for detecting anomalous fluxes in current transits based on comparison with the accumulated values,

FluxAlertUpdater for each preliminary alert gets all available historic transits,

FluxAlertCurator classifies alerts, decides which are not ongoing anymore, which to keep observing and which to publish.

These takers are to be run within the Ingestion stage of PhotPipe.

1.3 Assumptions/Limitations

In the current version (June 2009), the accumulated values are being taken from the same database as the new data. In the future the accumulated values will be read from previous cycle's database.

2 FluxAlertDetector

2.1 Taker

This taker takes `PhotFovTransit` and runs in time-interval mode.

2.1.1 Data Streams

Source Reader - reads sources for a given `TimeInterval` from Main Data Base.

Alert Table Manager - reads and writes to AlertTable (Auxiliary Data Stream)

2.1.2 Setup

Configures readers/writers for given job and reads in the data to memory.

Sources are read with the Source Reader and are stored for a quick retrieval in HashMap with a key being the SourceId (Long) and AlertSource as a value. DataAccessFailure exception is thrown if reading fails.

Alert Table (FluxAlertCandidate) is read and entries are stored in a HashMap with a key of SourceId (Long) and FluxAlertCandidate as a value. DataAccessFailure exception is thrown if reading fails.

Detection algorithm is instantiated and it can have the thresholds set here.

2.1.3 Take

PhotFovTransit is taken. It is assumed the transits were calibrated in an external run of du15.taker.IntCalApplier¹.

FluxAnomalyDetector.CheckTransit method is run on every transit. In case it fails, ProcessingFailedException is thrown and all hashMaps are cleared.

2.1.4 Teardown

After processing all transits for given TimeInterval the Alert Table is updated with all new and modified alerts. Calibration solver and maps storing the sources and alerts are cleared.

2.2 FluxAnomalyDetector

2.2.1 checkTransit method

This is the main method of the detection process. It finds the source observed in current transit in the map of sources for a given time interval and checks if the source from the current transit was alerting before. Based on the old alert status the following actions are undertaken:

¹The takers are run together in the job, e.g. `java -Xmx1G -jar dist/Testbed.6.2.jar -d Cycle3FV -l /Users/wyrzykow/WORK/GAIA/TESTBED/testbed3FVA -m gaia.cu5.dm.du15.dm.PhotFovTransit -t gaia.cu5.pipe.du15.takers.IntCalApplier, gaia.cu5.pipe.du17.takers.FluxAlertDetector`

No previous alert - check for anomaly is performed in *checkForAnomaly* method. If an anomaly found it is added to the alert map.

Source alerted before, but alert is not ongoing - check for anomaly is performed as above. If an anomaly is found, the alert entry in the alert map is overwritten with new one. If not, the old entry remains.

Source alerted before, alert is still ongoing, transit list is not empty - current transit is added to the list of transits in the alert and the entry for given source in the alert map is updated.

Source alerted before, alert is still ongoing, transit list is empty - nothing is done. This situation indicates the *FluxAlertUpdater* was not run yet for this alert.

2.2.2 *checkForAnomaly* method

This method returns *FluxAlertCandidate* with an alert if anomaly is detected and **null** if there is no anomaly. It runs the *DetectionAlgorithm* and if the anomaly is detected, the alert is created with the current source as an *AlertingSource*, alerting *transitId* and flag *OnGoing* set to true.

In case of a new source, the current transit is put in the list of transits in the alert. For such alert the *FluxAlertUpdater* will no be run.

In case of an existing source, the current transit is not added to the list, as it will be done by the *FluxAlertUpdater*, along with all other transits available for given source.

It is decided in the method *createAlert* whether the transit is being appended to the list or not.

2.2.3 CreateAlert

This method creates a new entry for an *alertTable*. If there was no source (**=null**), an empty source is created, i.e. with *sourceId* from the current transit and the accumulators equals to zero. In such case, the list of transits is created and the current transit is stored in it. It is an exception from regular running of the *Updater*, as here we assume this current transit is the first and only one observed for that source so far. If such source is observed once more, then, as usual, the transit will be stored by the *FluxAlertDetector* taker, as the list is not empty anymore.

>TODO> **TODO** Consider allowing the *Updater* reading all available transits in such cases?

If the source was not empty, the source with its accumulated values is stored in the created alert. The list of transits is created as empty (**=null**).

All alerts have their on-going flag set to true and the *alertingTransitId* is set to the current transit.

If there is a not null *DetectionAlgorithm* in the argument of the method, the threshold at which the detection was done is stored in the alert. The threshold is either `sigmaThreshold` (for existing sources) or `deltaMagThreshold` (for new sources).

2.3 Detection Algorithm

This class is responsible for testing for an anomaly. It compares previous accumulated observations (from `AlertSource`) with current flux measurement (from `PhotFovTransit`).

>TODO> **TODO** not coded yet: anomalies in colour (BP/RP). What about SM flux? what about astrometry?

2.3.1 Setup

Description of selected variables:

sigmaThreshold `double` is the main parameter of the detector, indicates the minimum value of difference between current flux and previous mean accumulated flux in units of accumulated `rms`. Default value = 3.0, but it can be also set in constructor.

source `AlertSource` stores accumulated values of flux, used for calculating mean and scatter. Can be set in constructor or later.

transit `PhotFovTransit` contains current transit with the values of fluxes to be tested for anomaly.

deviations `double[]` stores deviations (in sigmas) for each AF CCD

afsAboveTreshold `int[]` stores flags (0/1) if an AF CCD flux was deviating

nOfDeviatingAfs `int` number of deviating AF CCDs.

accumMeanFlux `double` mean flux, initialised with `calculateMean`.

accumScatter `double` scatter (*rms*), measure of the intrinsic variations in the accumulated data, used as a measure of deviations.

accumChiSq `double` χ^2/dof , returned from the source's method `getChiSq`

accumIsConstant `boolean` true if the accumulated (baseline) is constant. Initialised with `isAccumConstant` method.

Detection Algorithm uses the following properties:

gaia.cu5.pipe.du17.detection.defaultSigmaThreshold `double` value for `sigmaThreshold` (default 3.0)

gaia.cu5.pipe.du17.detection.detectionLimit `double` detection limit of Gaia in G band magnitudes, for SNe detections (default 20.0); for new sources the amplitude is measured from that level.

gaia.cu5.pipe.du17.detection.minNumberOfDeviants `int` minimum number of deviating AF CCDs required for alert to be triggered (default 3). If set to more than 1 helps to prevent alerting on cosmic rays.

gaia.cu5.pipe.du17.detection.deltaMagThreshold `double` minimum difference (in magnitudes) between current brightness and detection limit required to trigger on new sources (default 1).

gaia.cu5.pipe.du17.detection.minNumberOfAccums `short` minimum required number of accumulated observations (single AF+SM measurements) in the baseline to consider the source for testing for anomaly (default 5). Reasonable value is to have at least 3 transits, i.e. more than 21 observations.

gaia.cu5.pipe.du17.detection.chiSqConstantTreshold `double` threshold on χ^2/dof of the accumulated observations to consider the source was constant in the baseline. Default is 4, but it is good to relax it up to 50. Requires statistical fine-tuning.

There is a set of constructors of the Detector allowing for defining different `sigmaThreshold` than in the properties and for defining a reference source.

2.3.2 Detect

This is the main method for detecting the anomaly and it comes in different flavours. When called just with a transit, the source defined previously (either in constructor or with `setSource` method) is used. Other way of calling it is with transit and the source. In this case the source is set to be default for this instance of the Detector, so if the `detect` is called again just with a transit the last used source will be used as a reference.

If there is enough accumulated data in the source the main test is performed with `checkForAnomaliesOnCcds`. If the number of deviating datapoints in tested transit is larger than/equal to the required value (parameter `minNumberOfDeviants`) and the baseline is constant (variable `accumIsConstant`).

Additionally, the anomalous transit is also checked if all the deviations have the same sign, i.e. deviate in the same direction. If not, the anomaly is not considered.

>TODO> **TODO** Include such anomalies as well, as they might be varying on the timescales of one transit.

In principle, source can also be `null` (new source). In such case there is no requirement on the constant baseline ($\chi^2=0$) and the number of deviating ccds is derived based on magnitude deviations above the detection limit and `deltaMagThreshold`.

Finally, the direction of all deviations is checked if all of them are deviating in the same way. If not (e.g. in case of a variable star), the alert is not reported.

2.3.3 `checkForAnomaliesOnCcds`

These methods scan through all ccds (currently only AFs) and flux measured at each one is checked if it deviates from baseline or above the detection limit. Method `isDeviant` is called for each measurement and number of deviating ccd is gathered and returned.

2.3.4 `isDeviant`

This method checks each flux measurement for an anomaly.

If the source is new (indicated by zeroed number of accumulated observations or null source) deviation is significant if:

$$\text{Gaia_zeropoint} - 2.5 \log(\text{currentFlux}) < \text{detectionLimit} - \text{deltaMagThreshold}. \quad (1)$$

In case the source was observed before, the deviation is alerting if:

$$\left| \frac{\text{currentFlux} - \text{accumMeanFlux}}{\text{accumScatter}} \right| > \text{sigmaThreshold}. \quad (2)$$

Comparing absolute value of difference between current flux and accumulated mean allows for detection of deviations both up(e.g. supernovae) and down (e.g. transits or eclipses).

This method can be also called with different mean flux and scatter values.

2.3.5 `areDeviationsOfSameSign`

This method checks if all deviations found on various CCDs are of the same sign/direction. For example, the test will fail when some of deviations are less than -3 and the other are greater than 3, if the sigma threshold is set to 3.

3 FluxAlertUpdater

3.1 Taker

This taker takes `FluxAlertCandidate` and its main role is to read all transit available for given alert. Alert is updated in the Updated only if:

- is currently on-going, but has its transits list empty (alert was just created but `FluxAlertDetector`)

For the remaining alerts it is assumed the transits list is being appended with the new transits by the `FluxAlertDetector`.

3.1.1 Data Streams

Calibration Reader - reads the most recent calibrations from MDB

Transits Loader - reads all available transits for a given source

3.1.2 Setup

Configures readers/writers for given job and reads in the data to memory.

The most recent calibrations (`LsPhotCalSet`) are read. `DataAccessFailure` exception is thrown if reading fails. Solver (`LsPhotCalSolver`) is configured with this set of calibrations.

Transits Loader is configured.

3.1.3 Take

If the alert was not updated, i.e. its list of transits is not `null`, it means the alert is brand new and need updating. The list of transits is read with `loadTransits` method and is put into the alert object. The flag `hasNewData` is set to true to let curation know there are new measurements in the alert.

3.1.4 LoadTransits

This method reads the historic transits available for given source. The reader itself is configured in taker's `setup()`.

The reader reads `PhotFovTransit` class, which is then calibrated with the solver defined also in the setup. Then, calibrated transits are cast/converted to `AlertFovTransitDt` and stored in the list, which is returned by the method.

3.1.5 Teardown

Transit loader and calibration solver are cleared.

4 FluxAlertCurator

This taker takes `FluxAlertCandidate` as a main data stream and is run after `FluxAlertDetector` and `FluxAlertUpdater`. The main role of this part is to make a rough classification of the alerts and decide which ones are still ongoing (interesting), which have to be published immediately and which need more data.

4.1 Taker

4.1.1 Data Streams

AlertTable Manager - reads and writes to `AlertTable` (Auxiliary Data Stream)

4.1.2 Setup

From properties the parameters of classification are read:

detectionLimit Gaia's detection limit in magnitudes

4.1.3 Take

If an alert is ongoing and has new data since the last curation it is classified using separate class `Classifier`.

4.1.4 Teardown

After processing all alerts the Alert Table is overwritten with updated parameters of alerts.

4.2 Classifier - classification of alert

Simplified light curve is created using times, fluxes and their error bars from all gathered transits. The brightest (maximum) value is found and converted to magnitude.

First, the quality check on the alert is performed in method `checkAnomalyInSecondFov`. If this test is passed the main classification is performed. In case there are no observations prior the alert (new source) the amplitude (A) is calculated by subtracting `detectionLimit` parameter and the brightest value observed so far. In other cases, the baseline is derived as a mean brightness in the accumulated data and the amplitude is the difference between the baseline and maximum brightness magnitude.

The following classification is proposed.

1. **Supernovae** In the simplest situation a supernova is assumed to appear as a new source, which is indicated by zero number of accumulated observations in the `AlertSource` field of `FluxAlertCandidate`. The priority of such supernova alert is derived as dependend on the amplitude:

$$P = \left(1 - \frac{1}{\exp A}\right) * 100\% \quad (3)$$

2. **Dwarf nova** not coded yet

3. **Classical nova** not coded yet

4. **Microlensing** not coded yet

5. **Transit/Eclipse** not coded yet

99. **General Dip (transit)** If $A < 0$ the general decrease in brightness is detected. The priority is derived as for Supernova, based on the absolute value of A .

100. **General Bump** If $A > 0$ the general bump is detected. The priority is derived as for Supernova.

Once the alert was investigated and classified its flag `hasNewData` is set to `false`. This flag is switched off also in case of alerts which failed `checkAnomalyInSecondFov`. In such cases the alerts are switched off (ongoing flat to `false`)

>TODO> **TODO** Not coded yet: deciding whether the alert is passé.

The `Classifier` contains set of `static` methods returning the `byte` value of an alert class. It is advised to use these methods instead of the values, as they may change. Other solution to it would be to encode some enumeration for the classes, or define classes in properties.

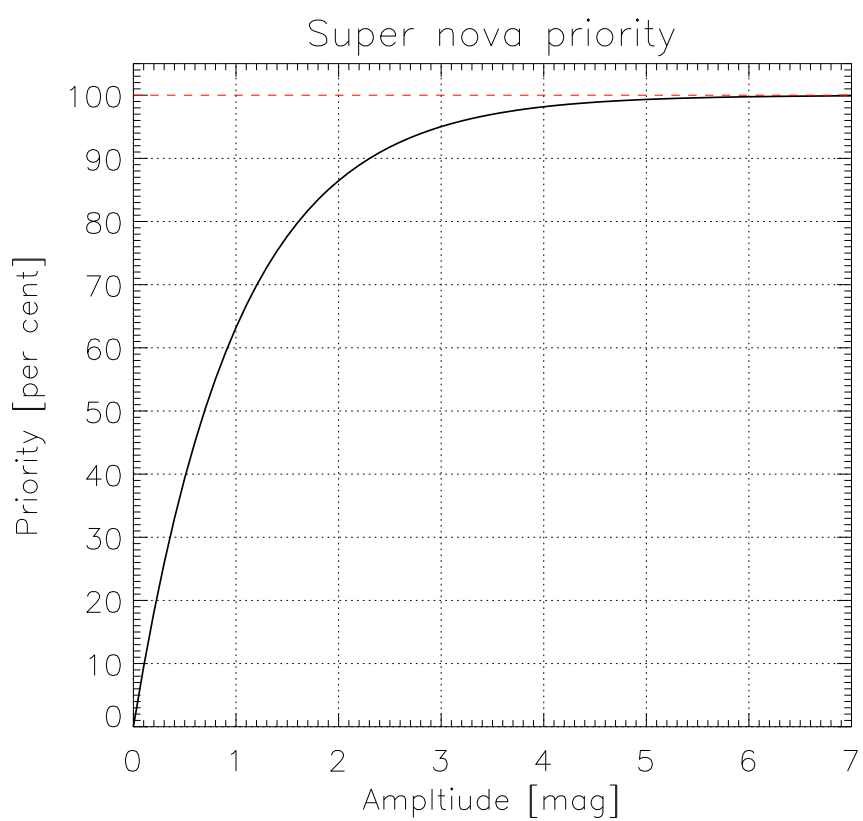


FIGURE 1: Priority function for Super novae in classification of FluxAlertCurator.

4.2.1 checkAnomalyInSecondFov

This test checks whether the alerting flux is visible on two subsequent field-of-view transits. First, the alerting transit is identified and the “next” transit is found. It is assumed the next transit is the following FoV transit, which requires the list of transits to be sorted according to their time. If the alerting transit occurred on the FOV=2, the check does not continue, as the next transit is probably very far from the alerting one. If the alerting transit is the last one in the list, the test does not continue.

If the next transit occurred in FOV=2 and was observed no later than time which it takes for Gaia to rotate the basic angle (around 6398.7s), this transit is tested on the detector if it also triggers an anomaly. If the alert is detected on the next transit the test is successful.

5 Alert Table

Alert Table is an Auxiliary Data Stream for DU17’s Flux Alerts and consists of set of items defined in class `FluxAlertCandidates` in DU17 Data Model (as in DM 2.1):

source `AlertSource` information on the alerting source along with data accumulated before the alert.

transits `List<AlertFovTransit>` appendable list of transits for storing all observations after the alert.

alertTransitId `long` id of the alerting transit.

hasNewTransits (Ongoing) `boolean` if true, transits are appended (means the alert is Ongoing)

classification `Short` simple classification index.

hasNewData `boolean` flag marking the alerts which had been updated with new data and require curation.

priority `short` “strength” of the alert, set during curation.

6 DU17’s Data Model

6.1 AlertFovTransitDt

This class is very similar to `PhotFovTransitDt`, although some fields irrelevant to DU17 were removed and a few useful ones were added.

Note this data type has very little information on the astrometry. In future it would be very useful to include the most accurate astrometry available in this type. This would allow the Flux Alerts to perform also astrometric alerts.

It consists of the following fields:

Solution Identifier `Integer*8` This is the identifier for the solution.

Source Identifier `Integer*8` This is the source identifier as defined in FDA-002.

Transit Identifier `Integer*8` This is the transit identifier as defined in BAS-017.
This is obtained from `cu3.idt.Interm.AstroElementary.transitId`.

SM CCD Transit `PhotCcdTransitDt` This is the G-band CCD transit photometry from the SM CCD.

AF CCD Transit `PhotCcdTransitDt*9` This is the main observed G-band CCD transit epoch photometry from the AF CCDs.

Integrated BP Transit `PhotCcdTransitDt` This and the following item can have the same structure as the G transits.

Integrated RP Transit `PhotCcdTransitDt`

AC Motion `Real*4 [rad/s]` The average AC motion of the source during the observation.
This will be used to reject fast moving Solar System objects Also see HV-001.

Status Flags `Integer*1` A combined flag, as in `PhotFovTransitDt`

velocity `Velocity[rad/s]`

6.1.1 AlertFovTransit

This class adds new methods and fields calculated on the fly to the `AlertFovTransitDt`. Some of them were copied from DU15's `PhotFovTransit`.

Note that the G-band Field Transit values should only be calculated *after* an internal calibration has been carried out.

Row `Integer*1` CCD row derived from `transitId`

Field of View `Integer*1` Number of the field of view (1 or 2)

Reference Time `Integer*4 [s]` Time of observation of the first CCD transit *i.e.* SM (Cycle 3) or AF1 (Cycle 4). This is derived from the `transitId` in `CycleUtil` and is returned in seconds.

Full Reference Time `Integer*8 [ns]` As Reference Time, but more accurate and returned in nanoseconds.

Heal Pixel `Integer*4` Heal pixel indicating position of the source. Derived from the `sourceId`.

Time `[s]` Transit time in seconds, calculated from Full Reference Time

TimeDays `[d]` Transit time in days, calculated from Full Reference Time

Mean G-band Flux `Real*4 [e-/s]` This is the mean G-band flux for a field transit over the CCD transit values. Mean and its error, should only be calculated *after* an internal calibration has been carried out.

Error of Mean G-band Flux `Real*4 [e-/s]` This is the error on the mean G-band flux for a field transit over the CCD transit values.

6.2 AlertSourceDt

This class is a Flux Alert's version of DU15's `PhotSourceDt`.

Note in this type there are two new fields `alpha` and `delta` to contain the Right Ascension and Declination of the source. It is not clear at the moment how they get filled.

It has the following fields:

Solution Identifier `Integer*8` Solution Identifier

Source Identifier `Integer*8` This is the source identifier as defined in FDA-002.
This is obtained from `cu3.idt.Source.sourceId`.

G-band accumulated data `AccumDataDt` Accumulated data for the internally calibrated G-band photometry from the AF CCDs.
This is not input data and is computed by CU5.

Integrated BP accumulated data `AccumDataDt` Accumulated data for the internally calibrated integrated BP photometry.
This is not input data and is computed by CU5.

Integrated RP accumulated data `AccumDataDt` Accumulated data for the internally calibrated integrated RP photometry.

This is not input data and is computed by CU5.

Alpha `Real*4` [**rad**] Right Ascension of the source

Delta `Real*4` [**rad**] Declination of the source

6.2.1 AlertSource

This data type expands `AlertSourceDt` type.

It first appears in the `du17.takers.FluxAlertDetector`, where it is read from the MDB as an Auxiliary Data Stream. It is the type which is taken by `du17.detection.DetectionAlgorithm`. Its accumulated values are used in the `DetectionAlgorithm`. Further on, in case of an Alert, this type is used to store the reference (baseline) values and colour, which are used in the `du17.takers.FluxAlertCurator`. Coordinates of the source from this data type will be used in the process of publishing the alert (not coded yet).

There is a constructor for this class allowing for easy conversion to `AlertSource` from `PhotSource`, but the source's coordinates `alpha` and `delta` in such case are set to 0.0.

G-band Mean Accumulated Flux `Real*4` Mean flux in G-band from all accumulated fluxes.

Error of G-band Mean Accumulated Flux `Real*4` Error of the mean flux in G-band from all accumulated fluxes.

G-band Flux Scatter `Real*4` Scatter of the accumulated fluxes. Equal to `Estimated RMS`.

Estimated RMS in G-band Accumulated Flux `Real*4` Estimated RMS of the accumulated fluxes.

$$rms = C_{rms} \sqrt{\frac{\chi^2}{dof} \langle \sigma \rangle}, \quad (4)$$

where

$$\langle \sigma \rangle = \sqrt{\frac{N-1}{\sum \frac{1}{\sigma^2}}}, \quad \frac{\chi^2}{dof} = UWV, \quad (5)$$

where `UWV` is the `UnitWeightVariance` obtained from the `AccumData` type, σ is the error measurement in flux, $\sum \frac{1}{\sigma^2}$ is the sum of weights, also obtained from `AccumData`. C_{rms} is a constant defined in the properties as `gaia.cu5.pipe.du17.detection.rmsCoeff`. Currently its value is set

to 1.0. Figure 2 shows ratio of estimated and real RMS for constant stars with $C_{rms} = 1.0$.

>TODO>

TODO check it everytime simulated data changes, or when also using SM fluxes.

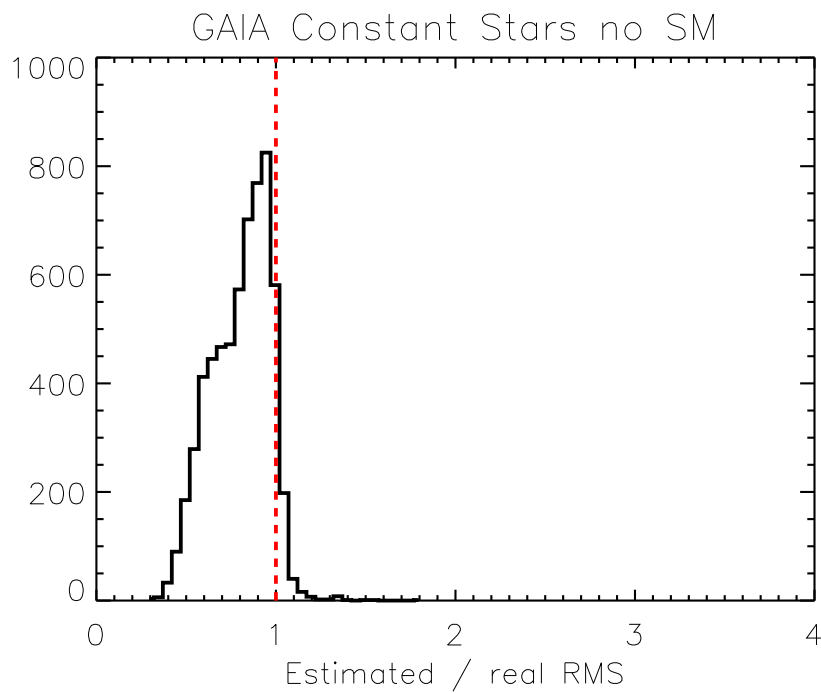


FIGURE 2: Ratio between estimated and real RMS derived for a testbed 3FV with the rmsCoeff=1.0 for constant stars

Chi Squared $\text{Real} \times 4$ Chi squared/dof of the accumulated G-band values (=UWV) indicating how constant is the baseline.

isConstant Flag `Boolean` G-band accumulated fluxes constancy derived as in the AccumData of DU15. Note in FluxAlerts the constancy is checked independently using a

threshold on Chi Squared.

Colour `Real*4 [mag]` Mean colour (BP-RP) from accumulated fluxes in BP and RP. Returned in magnitudes.

6.3 FluxAlertCandidateDt

This (and its expanded version `FluxAlertCandidate`) is the main class for storing alerts returned by the Flux Alerts.

It consists of the following fields:

Alert Transit Id `Integer*8` ID of the alerting transit.

Alert Source `AlertSource` Source with all its accumulated data *frozen* at the moment of the alert.

hasNewTransits `Boolean` Flag indicating the alert is on-going.

hasNewData `Boolean` Flag indicating if the candidate has new transits in the list which were not curated.

Classification `Integer*1` Classification of the alert candidate (value 0-255)

Alert Priority `Integer*1` Priority of the alert candidate (value 0-255)

Threshold `Real*2` Detection threshold during the alert detection.

Transits `List<AlertFovTransitDt>` List of internally calibrated transits. The list is first filled with all available transits by `FluxAlertUpdater`. After the alert it is appended by the `FluxAlertDetector` whenever the alerting source is observed.

solutionId Solution Identifier.

6.3.1 FluxAlertCandidate

Expansion of `FluxAlertCandidateDt` with handy methods.

Alerting Transit `AlertFovTransit` Entire transit which triggered the alert. Obtained from `alertTransitId`.

isOngoing Flag Boolean Overwrite flag to `hasNewTransits`. Indicates the alert is still on-going and will have all incoming transits appended.

>TODO> **TODO** Should be replaced in the DM.

Add Transit method (`AlertFovTransit`) Appends given transit to the list of transits collected so far. If the list is empty (**null**) it is created.

Source Id Integer*8 Identifier of the alerting source. Obtained from the `AlertSource` field.

Acronyms

The following table has been generated from the on-line Gaia acronym list:

Acronym	Description
AC	ACross scan (direction)
AF	Astrometric Field (in Astro)
BP	Blue Photometer
CCD	Charge-Coupled Device
DM	Data Model
FoV	Field of View (also denoted FOV)
ID	Identifier (Identification)
MDB	Main Database
RMS	Root-Mean-Square
RP	Red Photometer
SM	Star Mapper

References

- Bastian, U., 2006, *On-board AF/RVS cross-match indicator; transit label in MDB*,
GAIA-C3-TN-ARI-BAS-017,
URL <http://www.rssd.esa.int/llink/livelink/open/2795465>
- De Angeli, F., van Leeuwen, F., Hoar, J., et al., 2007, *Proposal for the object numbering scheme*,
GAIA-C1-MN-IOA-FDA-002,
URL <http://www.rssd.esa.int/llink/livelink/open/2698292>
- Voss, H., Jordi, C., Fabricius, C., et al., 2007, *AC flux loss analysis for AF and BP/RP*,
GAIA-C5-TN-UB-HV-001,
URL <http://www.rssd.esa.int/llink/livelink/open/2746204>